STAT 31512 Spring 2024 Scribe: Kexiang Wang
March 25, 2024 Lecturer: Frederic Koehler
These notes have not received the scrutiny of publication. They could be missing important references, etc.

Counting and Complexity Theory

1 Review of last time

Last time, we solved problem: for a given graph $G = (V, E)$, how to sample the spanning tree of this graph uniformly.

To solve this problem, we showed the that:

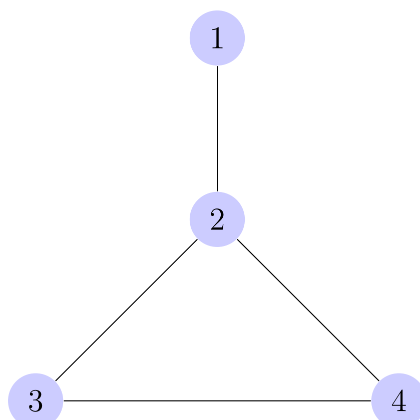
$$Z = det(L_{\sim 1, \sim 1})$$

where Z is the number of spanning trees of G and $L = D - A$ is the graph Laplacian.

But we actually solved a more general problem, where we could put weights on each edges. Let $\lambda_e \in \mathbb{R}_{\geq 0}$ be the weight of each edge. The we can define the probability measure for $S \subset E$:

$$\mathbb{P}_\lambda(S) = \frac{1}{Z_\lambda} 1_{\{\text{S is a spanning tree}\}} \cdot \prod_{e \in S} \lambda_e$$

Here is a example of a given graph:



We know that there are 3 different spanning trees in this graph. We can delete edge2,4, edge2,3 and edge 3,4 to get 3 different spanning trees. If we set $\lambda_{3,4} = 10^{100}$, and $\lambda_{2,3} = \lambda_{2,4} = 1$, we can find by the definition:

$$\mathbb{P}(\text{edge 3,4 is deleted in the tree}) \sim 10^{-100}$$

and

$$\mathbb{P}(\text{edge 2,3 is deleted in the tree}) = \mathbb{P}(\text{edge 2,4 is deleted in the tree}) \sim \frac{1}{2} - 10^{-100} \sim \frac{1}{2}$$

By similar argument, we can also show that

$$Z_\lambda = det(L_{\sim 1, \sim 1})$$

This means that we can compute Z_λ exactly in polynomial time. However, if we consider the general energy-based model for $x \in \{-1, 1\}^n$:

$$p(x) = \frac{1}{Z} exp(H(x))$$

Can we find a algorithm to compute Z in polynomial time? The answer is no. There are cases that we can compute $H(x)$ in polynomial time, but we cannot compute Z in polynomial time.

2 Introduction of complexity theory

Here, we introduce some basic concepts in complexity theory in compute science.

3-SAT: "NP-complete" problem:

Input: Formula in variable $(y_1, y_2, ..., y_n) \in \{0,1\}^n$, and clauses of boolean problem: i.e. $(y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_5) \vee ...$

Output: whether there exists y satisfying all clauses.

P is type of problems where we can solve the problem in polynomial time.

NP: Let $L \subset \{0,1\}^*$ be a arbitrary length bit string. L is in NP (non-deterministic polynomial time problem) if there exists polynomial time verifier $M(x,y) \in \{0,1\}$ such that:

1. $\forall x \in L$, there exists y polynomial size such that $M(x,y) = 1$
2. $\forall x \notin L$ and y polynomial size, $M(x,y) = 0$.

Note that $P \neq NP \Leftrightarrow 3SAT \notin P$.

#3-SAT problem: we explain it with a example:

Let $P_x(y) = \frac{1}{Z_x} 1_{\{\text{y satisfies x}\}}$, where x stands for the set of clauses of boolean problems as before and $y \in \{0,1\}^n$.

#3-SAT problems: Given x the clauses of boolean problems, compute Z_x is #3-SAT problems.

We can see that #3-SAT is at least as hard as 3-SAT problems. Since once we know if Z_x we know that if there exists y such that x is satisfied. i.e. $Z_x > 0$ implies the existence of such y, but knowing the result of 3-SAT problem is not enough to know the number of y such that x is satisfied.

$\#P$ problems: $f(x) \in \#P$ iff $f(x) = \#\{y : M(x,y) = 1\}$ can be computed in polynomial time.

FP problems is formally defined as follows:a binary relation $P(x,y)$ is in FP if and only if there is a deterministic polynomial time algorithm that, given x , either finds some y, such that $P(x,y)$ holds, or signals that no such y holds.

Remark: $P \neq NP \to FP \neq \#P$.

Consider the measure $P(x) = \frac{1}{Z} exp(H(x))$. It seems that counting Z is the same as sampling P.

But actually, for many sampling problems: computing Z is $\#P - hard$, but sampling P is polynomial time.

Example: No formula for Z, but sampling is possible.

Consider a graph $G = (V, E)$ graph, $\beta > 0$, $P_\beta(x) = \frac{1}{Z_\beta} exp(\beta \sum_{i \sim j} x_i x_j)$ where $x \in \{1, -1\}^n$.

There is a theorem by (Jerrom-Sindair/Swendsen-Wang) saying that we can sample this measure in polynomial time. This is non-trivial, but easier if $\beta < \frac{1}{max_i deg(i)}$

Theorem 1. *It is NP-hard to compute Z_β.*

Proof. Define $r = e^\beta$. Then we have

$$Z_\beta = \sum_x r^{\sum_{i \sim j} x_i x_j}$$

Noticing that $\sum_{i \sim j} x_i x_j \in [-\#edges, \#edges]$. Then can write z as

$$Z_\beta = \sum_{a=-|E|}^{a=|E|} b_a r^a \tag{1}$$

where b_a is the number of x such that $\sum_{i \sim j} x_i x_j = a$. Then by (1), we write Z_β as a polynomial of degree. Then by fundamental theorem of algebra, we can fix a polynomial by evaluating the polynomial on finitely many points, which takes polynomial time.

On the other hand, we know the fact: the Max-cut problem : $min_{x \in \{1,-1\}^n} \sum_{i \sim j} x_i x_j$ is NP-hard. The by computing Z_β for at least $n_2 + 1$ values of β, we can solve the Max-cut problem, which means that NP=P. But NP=P is the statement that people believe is false.

We will end this proof by providing an argument on why we can fix a polynomial by evaluating it at fintely many points.

Consider a polynomial of degree n:

$$P(x) = a_0 + a_1 x + ... + a_n x^n$$

If we value $P(x)$ at $n+1$ distinct points $x_0 ... x_n$ and we denote $y_i = P(x_i)$ for $i = 0, 1, 2...n$. Then we can get:

$$y = XA$$

where $y \in \mathbb{R}^{n+1}$ with entries are $y_i = P(x_i)$. $A \in \mathbb{R}^{n+1}$ with $A_i = a_i$ is the vector we want to solve. $X \in \mathbb{R}^{(n+1)\times(n+1)}$ is the Vandermonde matrix with $V_{i,j} = x_i^j$. We know that Vandermonde matrix is invertible since

$$det(X) = \prod_{0 \le i < j \le n} (x_i - x_j)$$

and $det(X) \neq 0$ since x_i are distinct. $\square$

Remark 1. The #Max-cut problem is actually #P-hard, and therefore, by the same argument, we can show that Z_β is #P-hard problem.